CA2 Assignment 2: Computer Vision with CNN

Data Management

Experimental Protocol

We implemented a stratified hold-out protocol for this experiment. The dataset, containing 11534 images across 10 unevenly distributed classes, was divided into three distinct sets: training (70%), validation (15%), and testing (15%). Crucially, we performed the split per class to maintain the original class distribution across all subsets, preventing representation bias. We used a fixed random seed (42) to ensure the reproducibility of the experiments. The final partition resulted in 8068 images for training, 1727 for validation and 1739 for testing. The identifiable class imbalance (ranging from 400 to 2000 images per class) was noted at this stage to inform the weighted loss function strategy.

Pre-processing and Data Augmentation

Pre-processing was applied to ensure input consistency and numerical stability. Although the source data is described as having a resolution of 64*64, to double-check, we performed a mandatory resizing operation to every image. This acts as a safeguard to guarantee strict dimensional consistency for the tensor operations, handling any potential outliers with incorrect sizes.

Moreover, images were converted to tensors and normalised. To prevent data leakage, the channel-wise mean and standard deviation were calculated exclusively from the training set. We calculated these statistics using a random sample of 1,000 images rather than the entire dataset. This sample size was chosen to balance statistical representation with computational cost during the data loading phase.

To prevent overfitting and improve the model's generalisation capabilities, we applied data augmentation dynamically to the training data. We utilised the following transformations:

- Random Horizontal Flip: (probability = 0.5) to simulate lateral variations in architectural views.
- Random Rotation: (±10 degrees) to account for minor camera tilts.
- Colour Jitter: (brightness=0.1, contrast=0.1) to introduce invariance to lighting conditions.

The validation and test sets were not augmented, undergoing only resizing and normalisation to provide a deterministic benchmark for evaluation.

Neural Networks

Proposed Network Architecture

We designed a custom Convolutional Neural Network (CNN) using Depthwise Separable Convolutions. This method is different from the standard convolutions, which perform spatial filtering and channel mixing simultaneously. This architecture works in 2 stages: a depthwise convolution (filtering each input channel independently) followed by a pointwise (1x1) convolution (combining the outputs). This reduces the computational cost and the number of trainable parameters compared to standard CNNs, improving efficiency without sacrificing representational depth. The network consists of three blocks, each followed by Batch Normalisation to accelerate convergence and ReLU activation for non-linearity. We used a Global Average Pooling layer at the final stage to generate the classification vector, eliminating the need for parameter-heavy fully connected layers found in traditional architectures.

Existing Architecture and Training Strategies ResNet18

We chose ResNet18 as the existing CNN benchmark. It utilises skip connections to mitigate the vanishing gradient problem, improving the training of deeper networks. To rigorously evaluate its performance, three distinct training strategies were implemented:

1. Training from Scratch: The model was initialised with random weights to establish a baseline for the architecture's capability to learn solely from the architectural dataset.
2. Feature Extraction: The convolutional base (pre-trained on ImageNet) was frozen, and only the final fully connected layer was trained. This evaluates the robustness of generic features (e.g. edges, textures) for this specific domain.
3. Fine-tuning: The model was initialised with ImageNet weights, but all layers were allowed to update. This enables the network to adapt high-level semantic features to the specific nuances of architectural styles (e.g., distinguishing a 'dome' from an 'apse').

Training Components

We used Weighted Cross-Entropy Loss to improve the imbalanced data. Class weights were calculated inversely proportional to class frequency, penalising the model more heavily for misclassifying under-represented classes (e.g., flying_buttress).

Two optimisers were selected based on the training context:

- SGD with Momentum (0.9): Used for the Customer CNN to ensure stable convergence and better generalisation during training from scratch.
- Adam: Employed for the ResNet variants due to its adaptive learning rate capabilities, which typically yield faster convergence for complex, pre-trained landscapes.

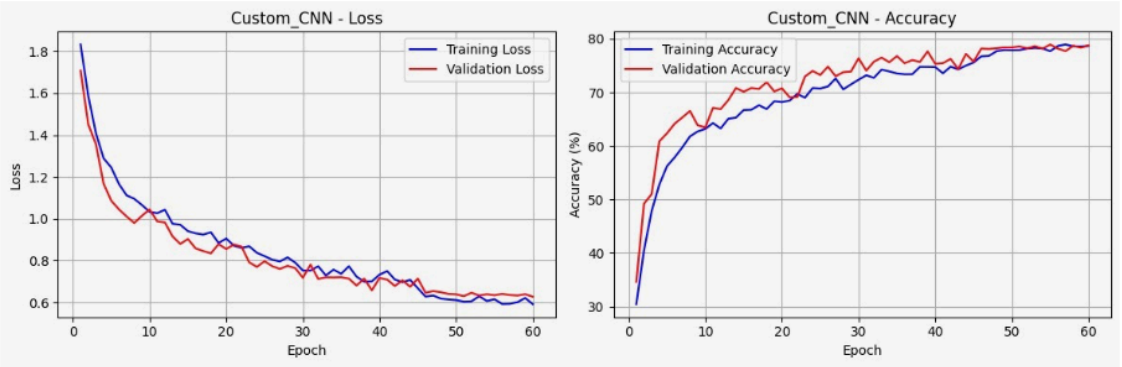
To prevent overfitting, Early Stopping (patience=7/10) and L2 Regularisation (weight decay) were utilised. Additionally, a ReduceLROnPlateau scheduler was included to automatically reduce the learning rate when validation loss stagnated, allowing the model to settle into finer minima.

Results and Discussion

The training results for the proposed and existing architectures are summarised below. Removing Early Stopping allowed the models to converge more fully, particularly improving 'Scratch', which needs more epochs to learn features from the ground up.

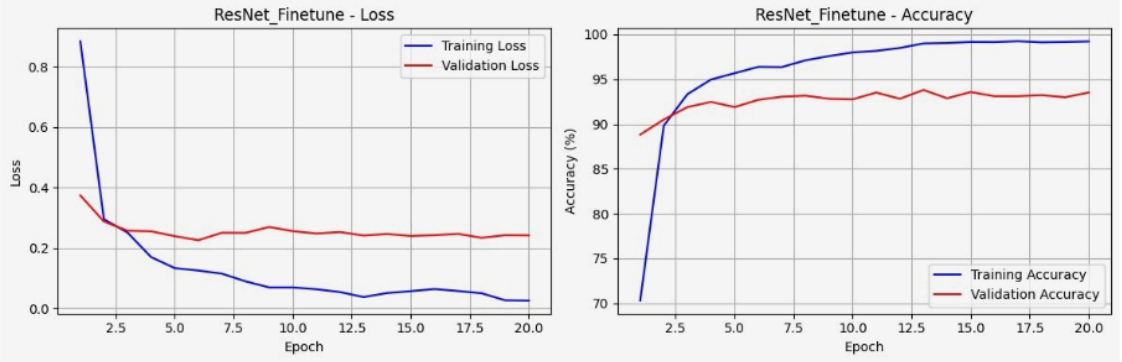
Model Strategy	Best Validation Accuracy
ResNet18 (Fine-tuning)	93.80%
ResNet18 (Scratch)	86.74%
Custom CNN	78.92%
ResNet18 (Feature Extraction)	75.45%

Custom CNN

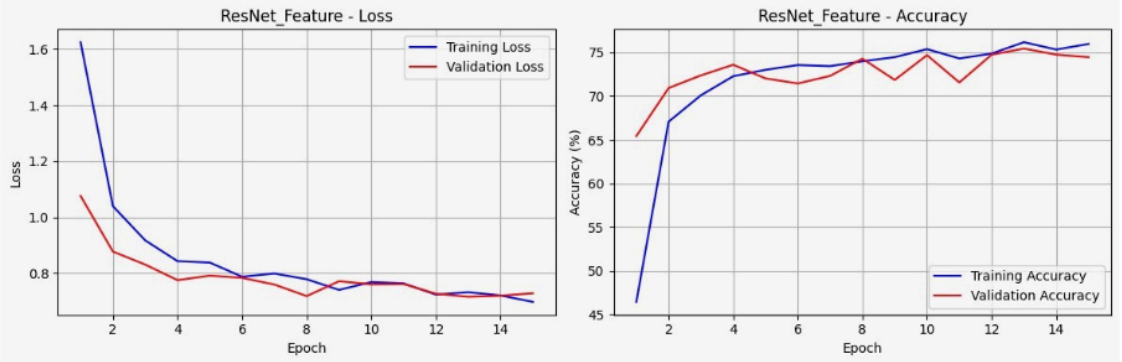


ResNet Scratch

ResNet Finetune



ResNet Feature Extraction



The ResNet18 (Fine-tuning) strategy proved superior (93.80%), validating the hypothesis that pre-training high-level features is very effective for architectural classification. Importantly, removing early stopping allowed the ResNet18 (Scratch) model to converge further, reaching 86.74%. Our Custom CNN achieved an accuracy of 78.92%, which achieved the design goal of efficiency. Despite having significantly fewer parameters, it outperformed the Feature Extraction strategy 75.45%. This suggests that while efficient, generic ImageNet features (Feature Extraction) are insufficient for this specific domain without fine-tuning weights. As the best performing model, the Fine-Tuned ResNet18 was selected for final evaluation.

Evaluate models

Test Set Performance

The best model (ResNet18 Fine-tuning) was evaluated on the held-out test set (1739 images) to provide an unbiased assessment of its generalisation capabilities. The model achieved a Test Accuracy of 92.58%, consistent with the validation accuracy (93.80%), showing no significant overfitting.

To capture performance beyond simple accuracy, we calculated multiple matrices to account for class imbalance:

- Precision (Weighted): 0.9253
- Recall (Weighted): 0.9258
- F1-Score (Weighted): 0.9254

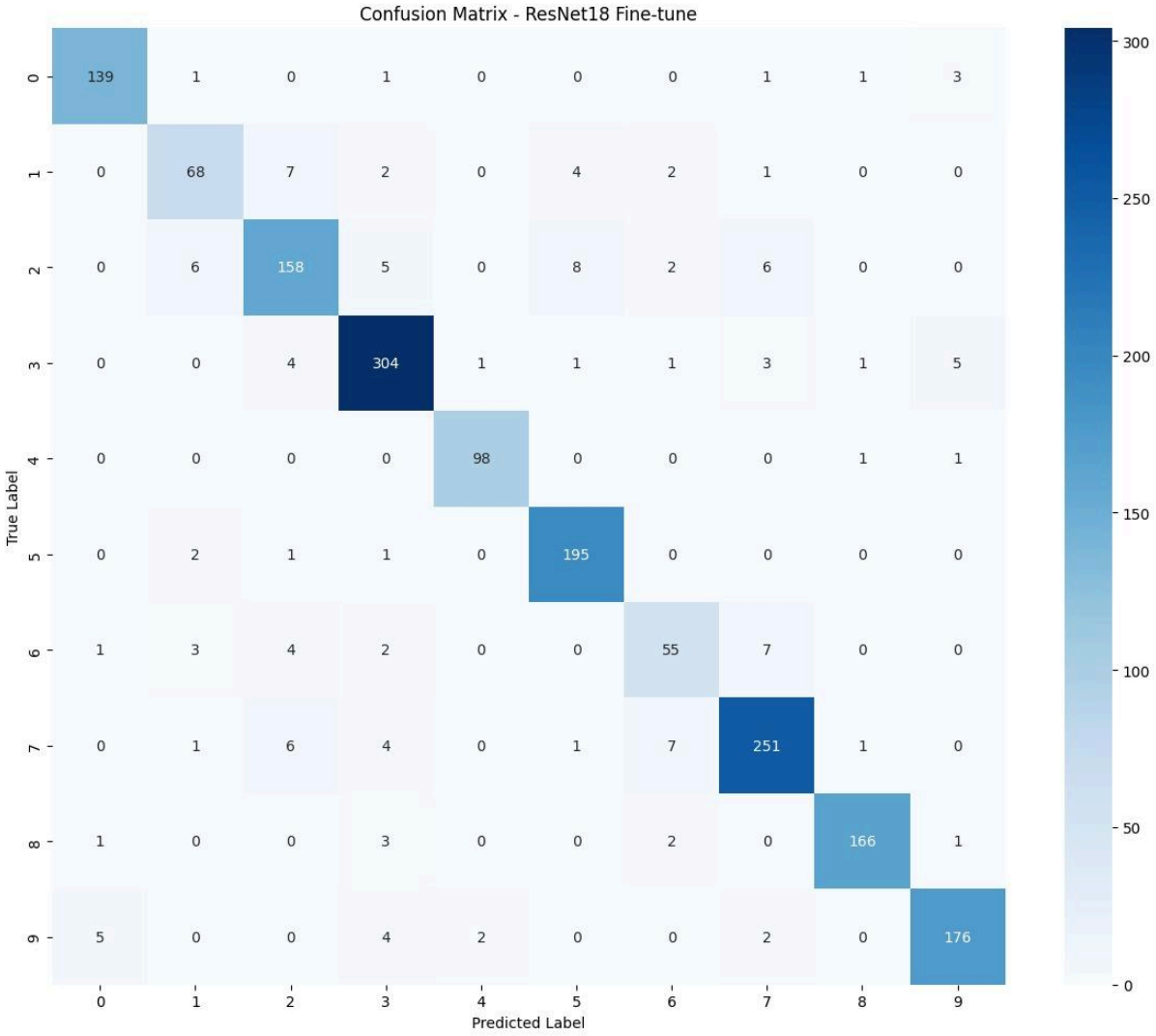
These scores confirm that the model is not biased towards the majority classes and is performing consistently across the dataset.

Class-wise Analysis

- Top performers: Dome (Inner) (Class 4) and Stained Glass (Class 8) achieved the highest F1-scores of 0.975 and 0.968, respectively. This suggests these architectural styles have distinct, high-contrast features like the light patterns of stained glass or the shape of domes that are easily distinguishable by the CNN.
- Worst performers: Flying Buttress (Class 6) proved the most difficult, with a Recall of 0.764 and an F1-score of 0.780. Apse (Class

l) also showed lower relative performance (Recall 0.810). These lower scores are likely due to visual ambiguity with other classes, possibly due to shared geometric structures.

Confusion Matrix Analysis



The confusion matrix visualises these classifications. The strong diagonal correlates with the high accuracy. However, notable off-diagonal clusters correspond to the lower metrics observed in Flying Buttress (Class 6). The model occasionally confuses Flying Buttress with Gargoyles (class 7). This is possibly due to shared textural patterns and camera angle. Despite these minor classifications, the model shows robust capability in differentiating the different classes.